

FLIP DISTANCE is in FPT time $\mathcal{O}(n + k \cdot c^k)$

Iyad Kanj

School of Computing, DePaul University
Chicago, USA
ikanj@cs.depaul.edu

Ge Xia

Dept. of Computer Science, Lafayette College
Easton, PA
xiag@lafayette.edu

Abstract

Let \mathcal{T} be a triangulation of a set \mathcal{P} of n points in the plane, and let e be an edge shared by two triangles in \mathcal{T} such that the quadrilateral Q formed by these two triangles is convex. A *flip* of e is the operation of replacing e by the other diagonal of Q to obtain a new triangulation of \mathcal{P} from \mathcal{T} . The *flip distance* between two triangulations of \mathcal{P} is the minimum number of flips needed to transform one triangulation into the other. The FLIP DISTANCE problem asks if the flip distance between two given triangulations of \mathcal{P} is k , for some given $k \in \mathbb{N}$. It is a fundamental and a challenging problem whose complexity for the case of triangulations of a convex polygon remains open for over 25 years.

In this paper we present an algorithm for the FLIP DISTANCE problem that runs in time $\mathcal{O}(n + k \cdot c^k)$, for $c = 392$, which implies that the problem is fixed-parameter tractable. The NP-hardness reduction for the FLIP DISTANCE problem given by Lubiw and Pathak can be used to show that, unless the exponential-time hypothesis (ETH) fails, the FLIP DISTANCE problem cannot be solved in time $\mathcal{O}^*(2^{o(k)})$. Therefore, one cannot expect an asymptotic improvement in the exponent of the running time of the presented algorithm.

1 Introduction

Let \mathcal{P} be a set of n points in the plane. A *triangulation* of \mathcal{P} is a partitioning of the convex hull of \mathcal{P} into triangles such that the set of vertices of the triangles in the triangulation is \mathcal{P} . Note that the convex hull of \mathcal{P} may contain points of \mathcal{P} in its interior.

A *flip* to an (interior) edge e in a triangulation of \mathcal{P} is the operation of replacing e by the other diagonal of the quadrilateral formed by the two triangles that share e , provided that this quadrilateral is convex; otherwise, flipping e is not permissible. The *flip distance* between two triangulations $\mathcal{T}_{initial}$ and \mathcal{T}_{final} of \mathcal{P} is the length of a shortest sequence of flips that transforms $\mathcal{T}_{initial}$ into \mathcal{T}_{final} . This distance is always well-defined and is $\mathcal{O}(|\mathcal{P}|^2)$ (e.g., see [8]). The FLIP DISTANCE problem is: Given two triangulation $\mathcal{T}_{initial}$ and \mathcal{T}_{final} of \mathcal{P} , and $k \in \mathbb{N}$, decide if the flip distance between $\mathcal{T}_{initial}$ and \mathcal{T}_{final} is k .

Triangulations are a very important subject of study in computational geometry, and they have applications in computer graphics, visualization, and geometric design (see [15, 17, 18, 22], to name a few). Flips in triangulations and the FLIP DISTANCE problem have received a large share of attention (see [3] for a review). The FLIP DISTANCE problem is a very fundamental and challenging problem, and different aspects of this problem have been studied, including the combinatorial, geometrical, topological, and computational aspects [1, 2, 3, 4, 7, 8, 10, 11, 16, 19, 20]. We can define the triangulations graph of \mathcal{P} , whose vertex-set is the set of all triangulations of \mathcal{P} , and in which two triangulations/vertices are adjacent if and only if their distance is 1. It is well-known that the triangulations graph has diameter $\mathcal{O}(n^2)$ [8], and hence, we can transform any

triangulation into another by a sequence of $O(n^2)$ flips. Moreover, it is known that the number of vertices in the triangulations graph is $\Omega(2.33^n)$ [1]. Therefore, solving the FLIP DISTANCE problem by finding a shortest path between the two triangulations in the triangulations graph is not feasible. A very similar problem to the FLIP DISTANCE was studied by Wagner [21] in 1936, who considered triangulated planar graphs instead.

The complexity of the FLIP DISTANCE problem was resolved very recently (2012) by Lubiw and Pathak [11, 12] who showed the problem to be NP-complete. Simultaneously, and independently, the problem was shown to be APX-hard by Pilz [16]. Very recently, Aichholzer et al. [2] showed the problem to be NP-complete for triangulations of a simple polygon. Resolving the complexity of the problem for the special case when \mathcal{P} is in a convex position (*i.e.*, triangulations of a convex polygon) remains a longstanding open problem for at least 25 years (see [20]); this problem is equivalent to the problem of computing the rotation distance between two rooted binary trees [4, 20]. Cleary and St. John [4] showed that this special case (convex polygon) is fixed-parameter tractable (FPT): They gave a kernel of size $5k$ for the problem and presented an $\mathcal{O}^*((5k)^k)$ -time¹ FPT algorithm based on this kernel. The upper bound on the kernel size for the convex case was subsequently improved to $2k$ by Lucas [13], who also gave an $\mathcal{O}^*(k^k)$ -time FPT algorithm for this case. We note that the kernelization approaches used in [4, 13] for the convex case are not applicable to the general case. In particular, the reduction rules used in [4, 13] to obtain a kernel for the convex case, and hence the FPT algorithms based on these kernels, do not generalize to the problem under consideration in this paper.

In this paper we present an $\mathcal{O}(n + k \cdot c^k)$ -time algorithm ($c = 392$) for the FLIP DISTANCE problem for triangulations of an arbitrary point-set in the plane, which shows that the problem is FPT. The NP-hardness reduction by Lubiw and Pathak can be used to show that, unless the exponential-time hypothesis (ETH) fails [9] (deemed to be very unlikely), the FLIP DISTANCE problem cannot be solved in time $\mathcal{O}^*(2^{o(k)})$. (The reduction is from the VERTEX COVER problem on cubic graphs, and results in a parameter value k for FLIP DISTANCE that is linear in the size of the instance of VERTEX COVER.) Therefore, one should not expect an asymptotic improvement in the exponent of the running time of the presented algorithm. While it is not very difficult to show that the FLIP DISTANCE problem is FPT based on some of the structural results in this paper, obtaining an $\mathcal{O}^*(c^k)$ -time algorithm, for some constant c , is quite involved, and requires a deep understanding of the underlying structure of the problem. We note that the (best) FPT algorithm for the convex case [13] runs in $\mathcal{O}^*(k^k)$ time.

Our approach is as follows. For any solution to a given instance of the problem, we can define a directed acyclic graph (DAG), whose nodes are the flips in the solution, that captures the dependency relation among the flips. We show that any topological sorting of this DAG corresponds to a valid solution of the instance. The difficult part is how, without knowing the DAG, to navigate the triangulation and perform the flips in an order that corresponds to a topological sorting of the DAG. We present a *very simple* nondeterministic algorithm that performs a sequence of “flip/move”-type local actions in a triangulation; each local action has constant-many choices. The key is to show that there exists such a sequence of actions that corresponds to a topological sorting of the DAG associated with a solution to the instance, and that the length of this sequence is linear in the number of nodes in the DAG. This will allow the nondeterministic algorithm to be simulated by an $\mathcal{O}^*(c^k)$ -time deterministic algorithm. To achieve the above goal, we develop structural results that reveal some of the intricacies of this fundamental and challenging problem.

Even though the triangulations considered in the paper are triangulations of a point-set in the plane, the presented algorithm works as well for triangulations of any polygonal region (even with

¹The $\mathcal{O}^*(\cdot)$ notation suppresses polynomial factors in the input size.

points in its interior). Moreover, using a reduction in [12] from the FLIP DISTANCE of triangulations of a polygonal region with holes to the FLIP DISTANCE of triangulations of a polygonal region with points in its interior, the algorithm presented in this paper can solve the (more general) FLIP DISTANCE problem of triangulations of a polygonal region with (possible) holes within the same time upper bound.

2 Preliminaries

Let \mathcal{P} be a set of n points in the plane, and let \mathcal{T} be a triangulation of \mathcal{P} . Let e be an interior (non-boundary) edge in \mathcal{T} . The *quadrilateral associated with e* in \mathcal{T} is defined to be the quadrilateral formed by the two adjacent triangles in \mathcal{T} that share e as an edge. Let e be an edge in \mathcal{T} such that the quadrilateral Q in \mathcal{T} associated with e is convex. A *flip f* with underlying edge e is an operation performed to e in triangulation \mathcal{T} that removes e and replaces it with the other diagonal of Q , thus obtaining a new triangulation of \mathcal{P} from \mathcal{T} . We use the notation $\epsilon(f)$ to denote the underlying edge e of a flip f in \mathcal{T} , and the notation $\phi(f)$ to denote the new diagonal/edge resulting from flip f . Note that $\phi(f)$ is not in \mathcal{T} . We say that a flip to an edge e is *admissible* in triangulation \mathcal{T} if e is in \mathcal{T} and the quadrilateral associated with e is convex. We say that two distinct edges e and e' in \mathcal{T} *share a triangle* if e and e' appear in the same triangle in \mathcal{T} . We say that two distinct edges e and e' between points in \mathcal{P} *cross* if e and e' intersect in their interior.

Let \mathcal{T} be a triangulation. A sequence of flips $F = \langle f_1, \dots, f_r \rangle$ is *valid* with respect to \mathcal{T} if there exist triangulations $\mathcal{T}_0, \dots, \mathcal{T}_r$ such that $\mathcal{T}_0 = \mathcal{T}$, f_i is admissible in \mathcal{T}_{i-1} , and performing flip f_i in \mathcal{T}_{i-1} results in triangulation \mathcal{T}_i , for $i = 1, \dots, r$. In this case we say that \mathcal{T}_r is the *outcome* of applying F to \mathcal{T} and that F *transforms* \mathcal{T} into \mathcal{T}_r , and we write $\mathcal{T} \xrightarrow{F} \mathcal{T}_r$. The *length* of F , denoted $|F|$, is the number of flips in it. Many flips in a sequence F may have the same underlying edge, but all those flips are distinct flips. For two flips f_i and f_h of F such that $i < h$, a flip f_p in F is said to be *between* f_i and f_h if $i < p < h$.

For two triangulations $\mathcal{T}_{initial}$ and \mathcal{T}_{final} of \mathcal{P} , the *flip distance* between $\mathcal{T}_{initial}$ and \mathcal{T}_{final} is the smallest $d \in \mathbb{N}$ such that there is a sequence F of length d satisfying that $\mathcal{T}_{initial} \xrightarrow{F} \mathcal{T}_{final}$. The FLIP DISTANCE problem is defined as follows:

FLIP DISTANCE

Given: Two triangulation $\mathcal{T}_{initial}$ and \mathcal{T}_{final} of \mathcal{P} .

Parameter: k .

Question: Is the flip distance between $\mathcal{T}_{initial}$ and \mathcal{T}_{final} equal to k ?

Let G be a graph. $V(G)$ and $E(G)$ denote the vertex-set and the edge-set of G , respectively, and $|G| = |V(G)| + |E(G)|$. For a directed graph G , a *weakly connected component* of G is a (maximal) connected component of the underlying undirected graph of G ; for simplicity, we will use the term *component* of a directed graph G to refer to a weakly connected component of G . Otherwise, we assume familiarity with basic graph theory, and refer to [5] for more information.

A *parameterized problem* is a set of instances of the form (x, k) , where x is the input instance and $k \in \mathbb{N}$ is the *parameter*. A parameterized problem is *fixed-parameter tractable*, shortly FPT, if there is an algorithm that solves the problem in time $f(k)|x|^c$, where f is a computable function and $c > 0$ is a constant. We refer to [6, 14] for more information about parameterized complexity.

3 Structural results

Let \mathcal{T} be a triangulation and let $F = \langle f_1, \dots, f_r \rangle$ be a valid sequence of flips with respect to \mathcal{T} . We denote by \mathcal{T}_i , for $i = 1, \dots, r$, the triangulation that is the outcome of applying the (valid) subsequence of flips $\langle f_1, \dots, f_i \rangle$ to \mathcal{T} .

Definition 1. Let f_i and f_j be two flips in F such that $1 \leq i < j \leq r$. Flip f_j is said to be *adjacent* to flip f_i , denoted $f_i \rightarrow f_j$, if:

- (1) either $\phi(f_i) = \epsilon(f_j)$ or $\phi(f_i)$ and $\epsilon(f_j)$ share a triangle in triangulation \mathcal{T}_{j-1} ; and
- (2) $\phi(f_i)$ is not flipped between f_i and f_j , that is, there does not exist a flip f_p in F , where $i < p < j$, such that $\epsilon(f_p) = \phi(f_i)$.

The above adjacency relation defined on the flips in F can be naturally represented by a directed acyclic graph (DAG), denoted \mathcal{D}_F , where the nodes of \mathcal{D}_F are the flips in F , and the arcs in \mathcal{D}_F represent the adjacencies in F . (Adjacency in the DAG is directed adjacency. That is, by saying that f_j is adjacent to f_i we mean that there is an arc from f_i to f_j .) Note that by definition, if $f_i \rightarrow f_j$ then $i < j$. For simplicity, we will label the nodes in \mathcal{D}_F with the labels of their corresponding flips in F . Observe that there are at most 4 edges that share a triangle with the underlying edge of any flip f_j in \mathcal{T}_{j-1} . Therefore, every node f_j in \mathcal{D}_F has indegree at most 5 (taking into consideration the possible last flip that created $\epsilon(f_j)$), and hence $|E(\mathcal{D}_F)| \leq 5 \cdot |V(\mathcal{D}_F)|$.

Lemma 1. Let \mathcal{T}_0 be a triangulation and let $F = \langle f_1, \dots, f_r \rangle$ be a sequence of flips such that $\mathcal{T}_0 \xrightarrow{F} \mathcal{T}_r$. Let $\pi(F)$ be a permutation of the flips in F such that $\pi(F)$ is a topological sorting of \mathcal{D}_F . Then $\pi(F)$ is a valid sequence of flips such that $\mathcal{T}_0 \xrightarrow{\pi(F)} \mathcal{T}_r$.

Proof. Proceed by induction on $|F|$. If $|F| \leq 1$, then the statement obviously holds true. Suppose that the statement is true for any F such that $|F| < r$, where $r > 1$, and consider a sequence F such that $|F| = r$.

Let f_s be the last flip in $\pi(F)$. Since $\pi(F)$ is a topological sorting of \mathcal{D}_F , f_s must be a sink in \mathcal{D}_F . It follows that no flip after f_s in F is adjacent to f_s in \mathcal{D}_F . Let Q be the quadrilateral associated with $\phi(f_s)$ in triangulation \mathcal{T}_s . Then no flips after f_s in F has its underlying edge in Q (i.e., as a boundary edge of Q or as a diagonal of Q), which means that the two adjacent triangles forming Q in \mathcal{T}_s remain unchanged throughout the flips after f_s in F . Therefore, we can safely move the flip f_s to the end of the sequence F without affecting the other flips in F nor the validity of F .

Let this new sequence be F' ; then it follows from the previous argument that $\mathcal{T}_0 \xrightarrow{F'} \mathcal{T}_r$. Since f_s appears at the end of F' , $F' - f_s$ is a valid sequence with respect to \mathcal{T}_0 that transforms \mathcal{T}_0 into some triangulation \mathcal{T} such that $\mathcal{T} \xrightarrow{f_s} \mathcal{T}_r$. Note that since f_s is a sink in \mathcal{D}_F , $\pi(F) - f_s$ is a permutation of the flips in $F' - f_s$ that is a topological sorting of $\mathcal{D}_F - f_s$. By the inductive hypothesis, $\pi(F) - \{f_s\}$ transforms \mathcal{T}_0 into \mathcal{T} . Since $\mathcal{T} \xrightarrow{f_s} \mathcal{T}_r$, appending f_s to the end of $\pi(F) - \{f_s\}$ results in $\pi(F)$ such that $\mathcal{T}_0 \xrightarrow{\pi(F)} \mathcal{T}_r$. This completes the inductive proof. \square

Corollary 1. Let \mathcal{T}_0 be a triangulation and let $F = \langle f_1, \dots, f_r \rangle$ be a sequence of flips such that $\mathcal{T}_0 \xrightarrow{F} \mathcal{T}_r$. For any given ordering (C_1, \dots, C_ℓ) of the components in \mathcal{D}_F , there is a permutation $\pi(F)$ of the flips in F such that $\mathcal{T}_0 \xrightarrow{\pi(F)} \mathcal{T}_r$, and such that for any two flips $f_i \in C_t$ and $f_j \in C_s$, where $1 \leq t < s \leq \ell$, f_i appears before f_j in $\pi(F)$. That is, all the flips in the same component appear as a consecutive block in $\pi(F)$, and the order of the blocks in $\pi(F)$ is the same as the given order of their corresponding components.

Proof. This follows from the fact that, for any given ordering of the components in \mathcal{D}_F , there is a topological sorting of \mathcal{D}_F in which all the flips in the same component of \mathcal{D}_F appear as a consecutive block, and in which the blocks appear in the same order as the given order of their components. \square

Definition 2. Let $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ be an instance of FLIP DISTANCE. An edge in $\mathcal{T}_{initial}$ that is not in \mathcal{T}_{final} is called a *changed edge*. If a sequence F is a solution to the instance $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$, we call a component in \mathcal{D}_F *essential* if the component contains a flip f such that $\epsilon(f)$ is a changed edge; otherwise, the component is called *nonessential*.

Lemma 2. Let $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ be an instance of FLIP DISTANCE, and suppose that F is a solution to the instance. Then every component of \mathcal{D}_F is essential.

Proof. Suppose, to get a contradiction, that \mathcal{D}_F contains a nonessential component C . Let F_C be the subsequence of F consisting of the flips that are in C . We will show that $F - F_C$ is a solution to the instance $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$. This will contradict the minimality of F because the number of flips in F is the flip distance between $\mathcal{T}_{initial}$ and \mathcal{T}_{final} .

By Corollary 1, we can assume that all the flips in F_C appear consecutively (*i.e.*, as a single block) at the end of F . Let \mathcal{T}' be the outcome of applying $F - F_C$ to $\mathcal{T}_{initial}$. It suffices to show that $\mathcal{T}' = \mathcal{T}_{final}$. Suppose that this is not the case. Since the number of edges in \mathcal{T}' and \mathcal{T}_{final} is the same, there must exist an edge $e \in \mathcal{T}'$ such that $e \notin \mathcal{T}_{final}$. Therefore, C must contain a flip f such that $\epsilon(f) = e$; we can assume that f is the first such flip in C . Since C is nonessential, $e \notin \mathcal{T}_{initial}$, otherwise e would be a changed edge. Therefore, there must exist a flip f' in $F - F_C$ such that $\phi(f') = e$; we can assume that f' is the last such flip in $F - F_C$. By the definition of adjacency in \mathcal{D}_F , there is an arc from node f' in $\mathcal{D}_F - C$ to node f in C , contradicting the assumption that C is a component of \mathcal{D}_F . This completes the proof. \square

Let $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ be an instance of FLIP DISTANCE, and suppose that F is a solution for $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$. By Lemma 2, \mathcal{D}_F does not contain nonessential components, and by Corollary 1, we can assume that all the flips in the same component of \mathcal{D}_F appear as a consecutive block in F . We shall call such a solution F satisfying the above properties a *normalized* solution. Suppose that $F = \langle f_1, \dots, f_k \rangle$ is a normalized solution to an instance $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k)$ of FLIP DISTANCE, and let C be a component of \mathcal{D}_F . We prove next a sequence of lemmas, combined into Lemma 7, that provide several sufficient conditions for a directed path to exist between two flips in C .

Lemma 3. Let f_i and f_h , where $i < h$ and $\epsilon(f_i) \neq \epsilon(f_h)$, be two flips in C such that $\phi(f_h)$ crosses $\epsilon(f_i)$, and $\epsilon(f_i)$ is not flipped between f_i and f_h . There is a directed path from f_i to f_h in C .

Proof. Assume that the statement is not true, and let f_i and f_h be the closest pair of flips in C (in terms of the number of flips between f_i and f_h) satisfying the conditions in the statement of the lemma and such that there is no directed path from f_i to f_h in C .

Consider the quadrilateral Q associated with $\phi(f_h)$ in \mathcal{T}_h . Since $\epsilon(f_i) \neq \epsilon(f_h)$ and $\phi(f_h)$ crosses $\epsilon(f_i)$, $\epsilon(f_i)$ cannot be a diagonal of Q or a boundary edge of Q . Because $\phi(f_h)$ crosses $\epsilon(f_i)$ and Q is devoid of points (of \mathcal{P}) in its interior, by a simple geometric observation, at least one boundary edge e of Q crosses $\epsilon(f_i)$. Therefore, there must exist a flip f_p , where $i \leq p < h$, such that $\phi(f_p) = e$. Since e is a boundary edge of Q which contains $\phi(f_h)$ as a diagonal in \mathcal{T}_h , e is also a boundary edge of Q in \mathcal{T}_{h-1} . This means that $\phi(f_p)$ and $\epsilon(f_h)$ share a triangle in \mathcal{T}_{h-1} , and hence there is an arc from f_p to f_h (we can assume that f_p is the last flip before f_h such that $e = \phi(f_p)$). If $i = p$ then there is a path (of length 0) from f_i to f_p ; on the other hand, if $i < p$, then since $\phi(f_p)$ crosses $\epsilon(f_i)$, $p < h$, and $\epsilon(f_p) \neq \epsilon(f_i)$, by the way f_h is chosen, there is a directed path from f_i to f_p in C . Therefore, there is a directed path from f_i to f_h in C — a contradiction. \square

Lemma 4. *Let f_i and f_h , where $i < h$, be two flips in C such that $\phi(f_h) = \epsilon(f_i)$, and $\epsilon(f_i)$ is not flipped between f_i and f_h . There is a directed path from f_i to f_h in C .*

Proof. Let f_p be the last flip between f_i and f_h such that $\phi(f_p) = \epsilon(f_h)$, and note that there is an arc from f_p to f_h in C . Then $\phi(f_p)$ (which is $\epsilon(f_h)$) crosses $\epsilon(f_i)$, and $\epsilon(f_i)$ is not flipped between f_i and f_p in C . Since $\epsilon(f_i)$ is not flipped between f_i and f_h , $\epsilon(f_p) \neq \epsilon(f_i)$. By Lemma 3, there is a directed path from f_i to f_p in C . Combining this directed path with the arc from f_p to f_h , we obtain a directed path from f_i to f_h in C . \square

Lemma 5. *Let f_i and f_h , where $i < h$, be two flips in C such that $\epsilon(f_i) = \epsilon(f_h)$. There is a directed path from f_i to f_h in C .*

Proof. Assume that the statement is not true, and let f_i and f_h be the closest pair of flips in C such that $\epsilon(f_i) = \epsilon(f_h)$ and there is no directed path from f_i to f_h in C . If $\epsilon(f_i) = \epsilon(f_p) = \epsilon(f_h)$ for some f_p between f_i and f_h , then by the way f_i and f_h are chosen, there is a directed path from f_i to f_p and a directed path from f_p to f_h , which implies that there is a directed path from f_i to f_h — a contradiction.

Now we can assume that $\epsilon(f_i)$ is not flipped between f_i and f_h . Let f_p be the last flip before f_h such that $\phi(f_p) = \epsilon(f_h)$; then there is an arc from f_p to f_h . Since $\phi(f_p) = \epsilon(f_h) = \epsilon(f_i)$, by Lemma 4, there is a directed path from f_i to f_p in C . Combining this directed path with the arc from f_p to f_h , we obtain a directed path from f_i to f_h in C — a contradiction. \square

Lemma 6. *Let f_i and f_h , where $i < h$, be two flips in C . If $\phi(f_i) = \epsilon(f_h)$, or if $\phi(f_i)$ and $\epsilon(f_h)$ share a triangle T in \mathcal{T}_j , for some j satisfying $i \leq j < h$, then there is a directed path from f_i to f_h in C .*

Proof. First consider the case when $\phi(f_i) = \epsilon(f_h)$. If $\phi(f_i) = \epsilon(f_h)$ is not flipped between f_i and f_h , then there is an arc from f_i to f_h in C . Suppose now that $\phi(f_i) = \epsilon(f_h)$ is flipped by another flip between f_i and f_h . Let f_p be the first such flip. Then there is an arc from f_i to f_p in C , and by Lemma 5, there is a directed path from f_p to f_h . Therefore there is a directed path from f_i to f_h in C .

Now consider the case where $\phi(f_i)$ and $\epsilon(f_h)$ share a triangle T in \mathcal{T}_j , for some j satisfying $i \leq j < h$, and assume, for the sake of a contradiction, that there is no directed path from f_i to f_h in C . Without loss of generality, assume that f_i and f_h are the closest such pair of flips in C .

Note that f_h cannot be the flip that immediately follows f_i , because in such case there would be an arc from f_i to f_h in C .

If $\phi(f_i)$ is flipped by a flip f_p satisfying $i < p \leq j$, then in this case $\phi(f_i)$ must be restored by a flip f_q where $p < q \leq j$ since $\phi(f_i)$ is in \mathcal{T}_j . We can assume that f_p and f_q are the first such flips. Therefore, there is an arc from f_i to f_p . Since $\phi(f_q) = \epsilon(f_p)$ and $\epsilon(f_p)$ is not flipped between f_p and f_q , by Lemma 4, there is a directed path from f_p to f_q in C . Since $\phi(f_q)$ (which is $\phi(f_i)$) and $\epsilon(f_h)$ share a triangle in \mathcal{T}_j , $q \leq j < h$, and the number of flips between f_q and f_h is less than that between f_i and f_h , by the way f_i and f_h are chosen, there is a path from f_q to f_h in C . Therefore, there is a directed path from f_i to f_h in C — a contradiction.

Now we can assume that $\phi(f_i)$ is not flipped by a flip f_p satisfying $i < p \leq j$. If no edge of the triangle T is flipped by a flip f_p between f_j and f_h , then there is an arc from f_i to f_h in C — a contradiction.

Finally, consider the case when an edge of T is flipped by f_p for the first time, where $j < p < h$. This means that $\phi(f_i)$ is in \mathcal{T}_{p-1} , and hence there is an arc from f_i to f_p . If $\epsilon(f_p) \neq \epsilon(f_h)$, then $\epsilon(f_p)$ and $\epsilon(f_h)$ share a triangle in \mathcal{T}_{p-1} , which means $\phi(f_p)$ and $\epsilon(f_h)$ share a triangle in \mathcal{T}_p . Since the number of flips between f_p and f_h is less than that between f_i and f_h , by the way f_i and f_h are

chosen, there is a path from f_p to f_h in C . Combining this path with the arc from f_i to f_p gives a directed path from f_i to f_h in C —a contradiction. On the other hand, if $\epsilon(f_p) = \epsilon(f_h)$, then by Lemma 5, there is a directed path from f_p to f_h . Combining this path with the arc from f_i to f_p gives a directed path from f_i to f_h in C —a contradiction. \square

Lemma 7. *Let f_i and f_h , where $i < h$, be two flips in C . If one the following conditions is true, then there is a directed path from f_i to f_h in C :*

- (1) $\phi(f_h)$ crosses $\epsilon(f_i)$.
- (2) $\phi(f_h) = \epsilon(f_i)$.
- (3) $\epsilon(f_i) = \epsilon(f_h)$.
- (4) $\phi(f_i) = \epsilon(f_h)$, or $\phi(f_i)$ and $\epsilon(f_h)$ share a triangle T in \mathcal{T}_j , for some j satisfying $i \leq j < h$.

Proof. This lemma follows from Lemmas 3, 4, 5, 6. Note that the condition in the statements of Lemmas 3, 4 that $\epsilon(f_i)$ is not flipped between f_i and f_h can be removed because of Lemma 5. \square

4 The algorithm

Using the structural results in Section 3, it is not difficult to obtain an FPT algorithm for FLIP DISTANCE that runs in $O^*(c^{k^2})$ time, for some constant c . Our goal, however, is to obtain an $O^*(c^k)$ -time algorithm for the problem, for some constant c . (Recall that even for the convex case the best FPT algorithm runs in $O^*(k^k)$ time [13].) Achieving this goal turns out to be quite challenging, and requires a deep understanding of the structure of the problem; we did so by analyzing the relation between the DAG that we associated with a solution to an instance of the problem and the changing structure of the underlying triangulations. We present our approach in this section.

4.1 Overview of the algorithm

In this subsection we give an intuitive description of how our algorithm works. Let $(\mathcal{T}_{\text{initial}}, \mathcal{T}_{\text{final}}, k)$ be an instance of FLIP DISTANCE. In order to solve the instance, the algorithm must perform a sequence of k flips that transforms $\mathcal{T}_{\text{initial}}$ into $\mathcal{T}_{\text{final}}$. By Lemma 1, it suffices for the algorithm to perform a sequence of flips that is a topological sorting of the DAG \mathcal{D}_F associated with a normalized solution F to the instance. Needless to say, the difficulty lies in the fact that we do not know F , nor do we know \mathcal{D}_F . By Lemma 2, each component of \mathcal{D}_F is essential, and hence, must contain a changed edge. The algorithm starts by picking a changed edge e in $\mathcal{T}_{\text{initial}}$ (which can be easily determined). Then there must exist a flip f in \mathcal{D}_F such that $e = \epsilon(f)$; let us refer to the component of \mathcal{D}_F containing f by C . We focus next on explaining how the algorithm, starting at e in $\mathcal{T}_{\text{initial}}$, performs a sequence of flips that is a topological sorting of C (Subsection 4.3), as this can be easily extended to a sequence of flips that is a topological sorting of the whole DAG \mathcal{D}_F (Subsection 4.4).

Clearly, the algorithm cannot start by flipping e , *i.e.*, performing f , since other flips may precede f in a solution. Instead, the algorithm searches for an edge $\epsilon(f_s)$ in $\mathcal{T}_{\text{initial}}$ that is the underlying edge of a source node f_s in C . The algorithm then flips $\epsilon(f_s)$ since flipping the underlying edge of a source node of C is safe. Now we explain how the algorithm searches for $\epsilon(f_s)$ in $\mathcal{T}_{\text{initial}}$ without having access to C . The algorithm starts at edge e in $\mathcal{T}_{\text{initial}}$ and “takes a walk” in which each step/action consists of moving to an edge that shares a triangle with the edge that the algorithm is currently at; the number of such local actions is the length of the walk. We show (Lemma 9) that there exists a source node f_s in C such that, starting at the changed edge e , the algorithm can

walk in the *current triangulation* $\mathcal{T}_{initial}$ from e to $\epsilon(f_s)$ such that the length of this walk is at most the length of the path from f_s to f in C . The reader should distinguish between adjacency in the DAG and adjacency in a specific triangulation. In particular, for two nodes that are adjacent in the DAG, their underlying edges may not even coexist in the current triangulation; so it is crucial that the walk can be done in the current triangulation. Now how does the algorithm walk from e to $\epsilon(f_s)$ in $\mathcal{T}_{initial}$? It does so nondeterministically by choosing, from the edge that it is currently located at (initially e), which adjacent edge to visit next. (This nondeterministic walk will be simulated deterministically by trying all such possible walks.) Suppose that the algorithm guessed the right walk nondeterministically and walked to $\epsilon(f_s)$ in $\mathcal{T}_{initial}$. The algorithm then flips $\epsilon(f_s)$ (thus performing flip f_s in C) to obtain a new triangulation $\mathcal{T}_{current}$, and stays at the edge $\phi(f_s)$ in $\mathcal{T}_{current}$. To continue the sequence of flips that corresponds to a topological sorting of C , the algorithm should flip next an edge in $\mathcal{T}_{current}$ that corresponds to a source node in the resulting DAG $C_{current} = C - f_s$. Therefore, the algorithm needs to take a walk from $\phi(f_s)$ in $\mathcal{T}_{current}$ to a source node in $C_{current}$, and flip the edge corresponding to that source node in $\mathcal{T}_{current}$, and so on. Therefore, ultimately, the algorithm is performing a sequence of nondeterministic walks and edge-flips. If the algorithm guesses each of these walks correctly, each walk will take the algorithm to an edge in the current triangulation corresponding to a source node in the current DAG, and the algorithm will flip that edge, and hence, eventually, the sequence of the flips performed by the algorithm will correspond to a topological sorting of C . To show how to perform this sequence of nondeterministic actions so that the total number of actions remains linear in k , we map the desired sequence of actions to a traversal of C (given in the procedure **Traversal** in Subsection 4.3), and upper bound the number of actions by the size of C , which is linear in k . Since the algorithm has no access to C , we will be using C only to prove the existence of a sequence of nondeterministic actions for the algorithm (described in Subsection 4.2), that can be mapped to a traversal of C in which the edge-flip actions correspond to a topological sorting of C .

Initially, the traversal starts at a node f in C corresponding to a changed edge e . We prove in Lemma 9 that there exists a path B in C from a source node f_s to node f that corresponds to a walk from e to $\epsilon(f_s)$ in the current triangulation upon entering C ; we call this path the *backbone* of C . The idea behind the notion of a backbone is to define a skeleton for the traversal of C , and to correspond the walks performed by the algorithm with this skeleton of C ensuring that the total number of actions performed by the algorithm is proportional to $|C|$. Now the algorithm walks in the current triangulation from e to $\epsilon(f_s)$ and flips $\epsilon(f_s)$. Note that to flip a single edge, the algorithm takes a walk in the current triangulation to a source node in C whose length (the walk) is upper bounded by the length of a corresponding path in C . Therefore, if we are not careful in how we do the traversal of C , the length of all these walks could be quadratic in k . Let $\mathcal{T}_{current}$ be the current triangulation, and let $C_{current} = C - f_s$. The removal of f_s from C may create several components in $C_{current}$. The actions of the algorithm are then mapped to traversals of each of these components recursively. We need to ensure that when the algorithm is done performing the flips in a component it can go back to proceed with the other components. To do so, the algorithm uses a stack to store the edge $\phi(f_s)$, resulting from flipping the “connecting node” f_s of all these components, so that the algorithm, after performing all the flips in a component of $C - f_s$, can go back by a single action to $\phi(f_s)$ (note that we cannot afford tracing our path back). Finally, we prove that all the backbones defined during the traversal of C form a forest (Lemma 10), allowing us to derive the desired upper bound on the total number of actions in the sequence.

4.2 The nondeterministic actions of the algorithm

The algorithm is a nondeterministic algorithm that starts from a changed edge in a triangulation $\mathcal{T}_{initial}$ and performs a sequence of actions. The algorithm is equipped with a stack. Each action σ of the algorithm acts on some edge e in a triangulation that we refer to as the *current triangulation* (before σ), denoted $\mathcal{T}_{current}$. Initially $\mathcal{T}_{current} = \mathcal{T}_{initial}$, and $\mathcal{T}_{current}$ before action σ is the triangulation resulting from applying the sequence of actions preceding σ to $\mathcal{T}_{initial}$. Each action σ of the algorithm is of the following possible types:

- (i) Move to one of the (at most 4) edges that share a triangle with e in $\mathcal{T}_{current}$.
- (ii) Flip e , and move to one of the 4 edges that shared a triangle with e in $\mathcal{T}_{current}$.
- (iii) Flip e , push the edge created by the flip into the stack, and move to one of the 4 edges that shared a triangle with e in $\mathcal{T}_{current}$.
- (iv) Flip e , jump to the edge on the top of the stack.
- (v) Flip e , jump to the edge on the top of the stack, and pop the stack.

A *walk* starting from an edge e in a triangulation is a sequence of actions all of which are of type (i).

Since there are 4 choices for each action of types (i)-(iii) and 1 choice for each action of types (iv)-(v), we have:

Proposition 1. *The number of possible choices for any action by the algorithm is at most 14.*

4.3 Traversal of a component of \mathcal{D}_F

Let $F = \langle f_1, \dots, f_k \rangle$ be a normalized solution to an instance of FLIP DISTANCE. Let C be a component of \mathcal{D}_F . By Corollary 1, we can assume that all the flips in C appear at the beginning of F , that is, form a prefix of F ; let $F_C = \langle f_1, \dots, f_t \rangle$ be the prefix of F corresponding to the flips in C . We will describe a recursive procedure **Traversal** that maps the nondeterministic actions performed by the algorithm to a traversal of C , such that the order of the flips performed by the algorithm corresponds to a topological sorting of C (repeated removal of source nodes by **Traversal**).

Initially, **Traversal** starts at a flip in C that corresponds to a changed edge at which the algorithm starts. During the recursion, **Traversal** removes nodes from C , which corresponds to the algorithm flipping the underlying edges of these removed nodes. Then **Traversal** is called recursively on components of the graph resulting from C after these nodes have been removed. Assume that the current triangulation is \mathcal{T} when **Traversal** is called on a component H . **Traversal** starts at a node f_h in H , called the *entry point* (defined in **Traversal**). We will show in Lemma 9 that there is a walk W that the algorithm can take in \mathcal{T} from $\epsilon(f_h)$ to $\epsilon(f_s)$, where f_s is a source node of H , such that W corresponds to a directed path $B = \langle b_1 = f_s, \dots, b_\ell = f_h \rangle$ in H ; we call B the *backbone* of H . The algorithm then flips $\epsilon(f_s)$, which corresponds to **Traversal** removing f_s from H , and then **Traversal** is recursively called on the components in $H - f_s$. During **Traversal**, for each component H that exists at this moment, we keep track of a simple path P in H that will ultimately allow us to prove that all the backbones form a forest.

Traversal(H, f_h, P)

1. If f_h is not a source node in H then the algorithm performs a walk W from $\epsilon(f_h)$ to $\epsilon(f_s)$ in \mathcal{T} , which corresponds to a (nontrivial) path/backbone $B = \langle f_s = b_1, \dots, b_\ell = f_h \rangle$, where f_s is a source node in H , and in this case we add B to P as a prefix ($P = B$ if $P = \emptyset$); otherwise (f_h is a source node), we let $f_s = f_h$;
2. The algorithm performs flip f_s in \mathcal{T} , which corresponds to removing f_s from H ; remove f_s from P and let f_b be the first flip in P after f_s is removed (in case f_b exists);
3. Let H_1, \dots, H_x be the components in H (note that f_s was removed). Define f_s to be the *connecting point* to each of the components H_1, \dots, H_x .

for $p = 1, \dots, x$ **do**:

- (a) **if** $x > 1$ and $p = 1$ **then** the algorithm nondeterministically pushes $\phi(f_s)$ into the stack;
- (b) **if** H_p does not contain any flip in P **then** define the *entry point* of H_p to be the flip $f_{h'}$ in H_p with the minimum index h' that is adjacent to f_s , and recursively correspond the actions of the algorithm to **Traversal**($H_p, f_{h'}, \emptyset$);
else (H_p contains a flip in P) define the entry point of H_p to be the flip $f_{h'}$ in H_p with the minimum index h' that is adjacent to f_s and that has a directed path P' to f_b (possibly $f_{h'} = f_b$); add P' to the P as a prefix, and recursively correspond the actions of the algorithm to **Traversal**($H_p, f_{h'}, P$);
- (c) **if** $x > 1$ and $p < x - 1$ **then** the algorithm nondeterministically moves to the top of the stack (which is $\phi(f_s)$);
- (d) **if** $x > 1$ and $p = x - 1$ **then** the algorithm nondeterministically moves to the top of the stack and pops the stack.

Let f_i, f_h be the connecting and entry points to a component $H \neq C$, respectively. At the top level of the recursion, where $H = C$ is a component of \mathcal{D}_F , define f_h to be a flip in C whose underlying edge $\epsilon(f_h)$ is a changed edge.

Lemma 8. *Let f_i, f_h be the connecting and entry points to a component $H \neq C$, respectively. Suppose that the current triangulation is \mathcal{T} when **Traversal** is called on H . Then $\phi(f_i)$ and $\epsilon(f_h)$ share a triangle in \mathcal{T} .*

Proof. Since there is an arc from f_i to f_h in H (by the way f_h was chosen), $\phi(f_i)$ and $\epsilon(f_h)$ share a triangle T in \mathcal{T}_{h-1} . We will show that T is in \mathcal{T} , which proves the statement of the lemma.

Suppose that this is not the case. Then there is a flip f_p in H between f_i and f_h that created T for the last time, and hence, both $\phi(f_i)$ and $\phi(f_p)$ belong to T in \mathcal{T}_p . Observe that $\phi(f_i) \neq \phi(f_p)$ because f_i has an arc to f_h (otherwise, f_i will not have an arc to f_h , but f_p will have instead). Therefore, $\phi(f_i)$ and $\phi(f_p)$ share triangle T in \mathcal{T}_p , and hence $\phi(f_i)$ and $\epsilon(f_p)$ also share some triangle in \mathcal{T}_{p-1} . This implies that f_i has an arc to f_p in H .

Since both $\phi(f_p)$ and $\epsilon(f_h)$ belong to triangle T (possibly identical) in \mathcal{T}_p , by Lemma 7, there is a directed path from f_p to f_h in the component C of \mathcal{D}_F . Since the nodes removed by **Traversal** are always source nodes, the path from f_p to f_h remains in the current component H (because f_p and f_h are in H). This, however, contradicts the choice of the entry point f_h of H (since f_i has an arc to f_p and there is a path from f_p to f_h). \square

Lemma 9. *Let f_h be the entry point to a component H . Suppose that the current triangulation is \mathcal{T} when **Traversal** is called on H . There is a walk W in \mathcal{T} from $\epsilon(f_h)$ to an edge $\epsilon(f_s)$, where f_s is a source node of H , such that both $\epsilon(f_h)$ and $\epsilon(f_s)$ are edges in \mathcal{T} , and there is a directed path B from f_s to f_h in H . Moreover, the length of the walk W is at most the number of nodes in B .*

Proof. If f_h is a source in H then $f_h = f_s$, the path B consists of f_s , and the length of the walk W is 0. The statement is trivially true. Now assume that f_h is not a source node in H .

By Lemma 8, $\epsilon(f_h)$ is an edge in \mathcal{T} . Let Q be the quadrilateral associated with $\epsilon(f_h)$ in \mathcal{T} . Since f_h is not a source in H and F is a minimal solution (under containment), one of the edges on the boundary of Q must be flipped before f_h ; let f_p be the first such flip in H . Since $\epsilon(f_p)$ and $\epsilon(f_h)$ share a triangle in \mathcal{T}_{p-1} , $\phi(f_p)$ and $\epsilon(f_h)$ share a triangle in \mathcal{T}_p , and by Lemma 7, there is a directed path from f_p to f_h in the component C of \mathcal{D}_F . Since the nodes removed by **Traversal** are always source nodes, there is a directed path from f_p to f_h in the current component H . The edges $\epsilon(f_h)$ and $\epsilon(f_p)$ share a triangle in \mathcal{T} , and hence, in one action (of type (i)) the algorithm can go from $\epsilon(f_p)$ to $\epsilon(f_h)$ in \mathcal{T} .

If f_p is a source node in H , then we are done; otherwise, applying the above argument to f_p , we can find a flip f_q such that $\epsilon(f_p)$ and $\epsilon(f_q)$ share a triangle in \mathcal{T} and there is a directed path from f_q to f_p in H . We can repeat this process until we reach a source node f_s in H . Going from $\epsilon(f_h)$ to $\epsilon(f_s)$ in \mathcal{T} involves only actions of type (i), and hence, define a walk W from $\epsilon(f_h)$ to $\epsilon(f_s)$ in \mathcal{T} . The length of W is at most the total number of flips in a directed path B from f_s to f_h in H , which is composed of the directed paths defined in the process described above (from f_p to f_h , f_q to f_p , and so on). \square

Lemma 10. *The backbones defined during the **Traversal** of C are edge-disjoint and the underlying (undirected) graph formed by them is a forest.*

Proof. The backbones of C can be ordered by the recursive calls to **Traversal** that defined them. Proceed by contradiction. Let B and B' be two backbones that were formed during the call to **Traversal** on C such that B was formed before B' , and assume that B' shares edges with B . Suppose that B was defined as the backbone of some component H when **Traversal** was called on H , and let f_s be the source node of B . After finding B , and before making any subsequent recursive calls, **Traversal** adds B as a prefix to some path P . Afterwards, **Traversal** recurses on each component of $H - f_s$. Because P contains B , B' must be contained in the component of $H - f_s$ that contains some node in P ; otherwise, B' would not share any edges with B . Moreover, from the way **Traversal** works, it must be the case that B' was appended as a prefix to a directed path P' that contains the edges of B that remained when B' was defined. Since B and B' cannot share any edge that was removed from B , B and B' must share an edge of P' . This, however, contradicts the acyclicity of C , because in this case appending B' to P' would form a directed cycle. Finally, it follows by the same argument that the only node that B' can share with B is the last node (sink) of B' . This shows that the underlying (undirected) graph of all the backbones is a forest. \square

Theorem 1. *Let C be a component of \mathcal{D}_F . There is a sequence of actions for the nondeterministic algorithm of length at most $2|V(C)|$ such that, when started from a changed edge $\epsilon(f_h)$ for some $f_h \in C$, performs the flips in C in a topologically-sorted order.*

Proof. It suffices to show that the sequence of steps performed by **Traversal** on C corresponds to a sequence of actions by the algorithm of length at most $2|V(C)|$. The order of the flips performed by the algorithm in this sequence corresponds to a topological sorting of C because every flip corresponds to the removal of a source node of the resulting DAG by **Traversal**.

Traversal starts in a component H at an entry node f_h . In Lemma 9 we showed that there is a path $B = \langle f_s = b_1, \dots, b_\ell = f_h \rangle$ from a source node f_s in H to f_h that corresponds to a walk by the algorithm in the current triangulation \mathcal{T} from edge $\epsilon(f_h)$ to $\epsilon(f_s)$; moreover, the length of this walk is at most the number of flips in B . The algorithm can perform this walk using actions of type (i), and the number of such actions is at most the length of B . Now the algorithm is at edge $\epsilon(f_s)$ in \mathcal{T} . **Traversal** then removes the source node f_s . This corresponds to flipping edge $\epsilon(f_s)$, which is one action of the algorithm either of type (ii) or (iii). Next, **Traversal** considers every component H_p , $p = 1, \dots, x$, of $H - f_s$. For each component H_p , **Traversal** finds an entry point $f_{h'}$ to H_p , and recursively traverses H_p starting from $f_{h'}$. In Lemma 8, we showed that the edges $\phi(f_s)$ and $\epsilon(f_{h'})$ share a triangle in the triangulation when **Traversal** is recursively called on H_p . If there is only one component in $H - f_s$, there is no need for stack operations. If there is more than one component in $H - f_s$, we push $\phi(f_s)$ into the stack after flipping f_s . In case there is only one component left, we also pop the stack after jumping to the top of the stack. It is not difficult to see that each of the steps of **Traversal** corresponds to one action of the algorithm of types (i)-(v).

It follows from the above that the sequence of steps performed by **Traversal** corresponds to a sequence of actions by the nondeterministic algorithm. The actions of the algorithm can be classified into two categories: actions with flips and actions without flips. The number of actions with flips is at most the number of nodes $|V(C)|$ in C . The actions without flips are of type (i) and they can be further divided into two groups: (a) those done in the walks taken by the algorithm along the backbones, and (b) those done by the algorithm when it moves from the edge on the top of the stack to a new component H_p , $p = 2, \dots, x$, of $H - f_s$. The number of actions in group (a) is at most the total length of all the backbones formed during the call to **Traversal** on C , which is at most the number of the edges in the forest defined in Lemma 10. To upper bound the number of actions in group (b), observe that $x - 1$ such actions are needed for the components H_2, \dots, H_x , which is at most the number of new trees created by **Traversal** in the components of $H - f_s$ not containing B . This means that the number of actions in group (B) is at most the number of trees in the forest defined in Lemma 10. It follows that the total number of actions without flips is at most $|V(C)|$, and hence, the total number of actions needed by the algorithm is at most $2|V(C)|$. \square

4.4 Putting all together: the whole algorithm

Let F be a solution of the instance $(\mathcal{T}_{\text{initial}}, \mathcal{T}_{\text{final}}, k)$. By Corollary 1 and Lemma 2, we can assume that F is normalized. Order the changed edges arbitrarily; let the ordering be \mathcal{O} . The algorithm starts by guessing the number of components in \mathcal{D}_F ; let this number be $t \leq k$. The algorithm then guesses the number of flips k_1, \dots, k_t in the components C_1, \dots, C_t , respectively, of \mathcal{D}_F satisfying $k_1, \dots, k_t \geq 1$ and $k_1 + k_2 + \dots + k_t = k$. Fix such a guess (k_1, \dots, k_t) .

The algorithm performs t iterations: $\ell = 1, \dots, t$. We define $\mathcal{T}_{\text{initial}}^\ell$, $\ell = 1, \dots, t$, to be the triangulation that resulted from $\mathcal{T}_{\text{initial}}$ after the flips in the first ℓ iterations are performed. We define $\mathcal{T}_{\text{initial}}^0 = \mathcal{T}_{\text{initial}}$. For each $\ell = 1, \dots, t$, do the following: Pick the next edge $e \in \mathcal{O}$. If e is not a changed edge anymore with respect to $\mathcal{T}_{\text{initial}}^{\ell-1}$ and $\mathcal{T}_{\text{final}}$, then skip to the next edge in \mathcal{O} . Otherwise (e is in $\mathcal{T}_{\text{initial}}^{\ell-1}$ but not in $\mathcal{T}_{\text{final}}$), perform a sequence of actions starting from e in $\mathcal{T}_{\text{initial}}^{\ell-1}$ until either the number of flips performed is k_ℓ , or the number of actions performed reaches $2k_\ell$. Let F_ℓ be the sequence of flips performed in the current iteration, and note that $\mathcal{T}_{\text{initial}}^{\ell-1} \xrightarrow{F_\ell} \mathcal{T}_{\text{initial}}^\ell$. After the last iteration $\ell = t$, if $\mathcal{T}_{\text{initial}}^t = \mathcal{T}_{\text{final}}$ then accept; otherwise reject.

Theorem 2. *Let $(\mathcal{T}_{\text{initial}}, \mathcal{T}_{\text{final}}, k)$ be an instance of FLIP DISTANCE. The above nondeterministic algorithm decides $(\mathcal{T}_{\text{initial}}, \mathcal{T}_{\text{final}}, k)$ correctly, and it can be simulated by a deterministic algorithm that runs in time $\mathcal{O}(n + k \cdot 392^k)$.*

Proof. It is easy to see that the correctness of the algorithm follows from the following: (1) there is a guess for the algorithm of the correct number of components t , and of (k_1, \dots, k_t) such that k_i is the exact number of flips in C_i , $i = 1, \dots, t$; and (2) by Theorem 1, there is a nondeterministic sequence of actions by the algorithm of length at most $2k_i$ that, starting from a changed edge in C_i , performs the k_i flips in C_i in a topologically-sorted order.

We only need to analyze the deterministic running time needed to simulate the nondeterministic algorithm. The initial processing of the triangulations to find the changed edges takes $\mathcal{O}(n)$ time. The total number of sequences (k_1, \dots, k_t) , for $t = 1, \dots, k$, satisfying $k_1 + \dots + k_t = k$ and $k_1, \dots, k_t \geq 1$, is known as the *composition number* of (integer) k , and is equal to 2^{k-1} . For each such sequence (k_1, \dots, k_t) , we iterate through the numbers k_1, \dots, k_t in the sequence. For a number k_i , $1 \leq i \leq t$, by Theorem 1, we need to try every sequence of at most $2k_i$ actions, and in which each action is one of 14 choices (by Proposition 1). Therefore, the number of such sequences is at most 14^{2k_i} . It follows that the total number of sequences that the algorithm needs to enumerate to find a witness to the solution (if it exists) is at most: $\sum_{t=1}^k \sum_{(k_1, \dots, k_t)} (14^{2k_1} \times \dots \times 14^{2k_t}) = \mathcal{O}(2^{k-1} 14^{2k}) = \mathcal{O}(392^k)$.

Since each sequence of actions can be carried out in time $\mathcal{O}(k)$, and the resulting triangulation at the end of the sequence can be compared to \mathcal{T}_{final} in $\mathcal{O}(k)$ time as well (by a careful implementation that keeps track of the edges that have been modified along the way and the triangles containing them), the running time for each enumerated sequence is $\mathcal{O}(k)$. It follows from the above that the running of the deterministic algorithm is $\mathcal{O}(n + k \cdot 392^k)$. Finally we point out that the algorithm needs to decide whether k is the flip distance between $\mathcal{T}_{initial}$ and \mathcal{T}_{final} , which means that no sequence of flips of length smaller than k exists that transforms $\mathcal{T}_{initial}$ to \mathcal{T}_{final} . This can be decided by invoking the algorithm on each of the instances $(\mathcal{T}_{initial}, \mathcal{T}_{final}, k')$, for $k' = 0, \dots, k$. The running time remains $\mathcal{O}(n + k \cdot 392^k)$ because $\sum_{k'=0}^k \mathcal{O}(k' \cdot 392^{k'}) = \mathcal{O}(k \cdot 392^k)$ (note that computing the changed edges is done once). \square

References

- [1] O. Aichholzer, F. Hurtado, and M. Noy. A lower bound on the number of triangulations of planar point sets. *Computational Geometry*, 29(2):135–145, 2004.
- [2] O. Aichholzer, W. Mulzer, and A. Pilz. Flip distance between triangulations of a simple polygon is NP-complete. In *proceedings of ESA*, volume 8125 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2013.
- [3] P. Bose and F. Hurtado. Flips in planar graphs. *Computational Geometry*, 42(1):60–80, 2009.
- [4] S. Cleary and K. St. John. Rotation distance is fixed-parameter tractable. *Information Processing Letters*, 109(16):918–922, 2009.
- [5] R. Diestel. *Graph Theory*. Springer, Berlin, 4th edition, 2010.
- [6] R. Downey and M. Fellows. *Parameterized Complexity*. Springer, New York, 1999.
- [7] S. Hanke, T. Ottmann, and S. Schuierer. The edge-flipping distance of triangulations. *Journal of Universal Computer Science*, 2(8):570–579, 1996.
- [8] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, 22(3):333–346, 1999.

- [9] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
- [10] C. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972.
- [11] A. Lubiw and V. Pathak. Flip distance between two triangulations of a point set is NP-complete. In *proceedings of CCCG*, pages 119–124, 2012.
- [12] A. Lubiw and V. Pathak. Flip distance between two triangulations of a point set is NP-complete. arXiv.org e-Print archive, paper cs.CG/1205.2425, May 2012.
- [13] J. Lucas. An improved kernel size for rotation distance in binary trees. *Information Processing Letters*, 110(12):481–484, 2010.
- [14] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, USA, 2006.
- [15] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, New York, NY, USA, 1992.
- [16] A. Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014.
- [17] Alan Saalfeld. Joint triangulations and triangulation maps. In *proceedings of SoCG*, pages 195–204. ACM, 1987.
- [18] L. Schumaker. Triangulations in CAGD. *IEEE Computer Graphics and Applications*, 13(1):47–52, 1993.
- [19] R. Sibson. Locally equiangular triangulations. *The Computer Journal*, 21(3):243–245, 1978.
- [20] D. Sleator, R. Tarjan, and W. Thurston. Rotation distance, triangulations, and hyperbolic geometry. In *proceedings of STOC*, pages 122–135. ACM, 1986.
- [21] K. Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.
- [22] D. Watson and G. Philip. Systematic triangulations. *Computer Vision, Graphics, and Image Processing*, 22(2):310, 1983.